# 2017년 프로그래밍언어 개론

## -04-

| 제출일자 | 2017.04.12. |
|---|---|
| 이 름 | 정 윤 수 |
| 학 번 | 201302482 |
| 분 반 | 03 |

## def __str__(self):

```
def __str__(self):
    result = ''

    if self.type is TokenType.ID:
        result = '[' + NODETYPE_NAMES[self.type] + ':' + self.value + ']'
    elif self.type is TokenType.INT:
        result = '[' + NODETYPE_NAMES[self.type] + ':' + self.value + ']'
    elif self.type is TokenType.LIST:
        result = '('+str(self.value)+')'
    else:
        result = '['+NODETYPE_NAMES[self.type]+']'
    if self.next is None:
        return result
    else:
        return result+" "+str(self.next)
    # fill out
    # next 노드에 대해서도 출력하도록 작성
    # recursion 이용
```

문제 해결 방법

- str()함수는 Node클래스의 객체가 print()함수로 인해서 출력이 되어질 때 호출이 되어지는 함수이다. 여기 에서는 self의 type을 보고 result에 출력을 할 내용을 저장을 한다. type이 ID 또는 INT이면 type과 value를 둘다 출력을 해주고 ' ( ' 이면 List라는 것 임으로 value값을 이용을 하여 다시 str()함수를 호출을 한다. 다른 문자이면 type만을 result에 출력 내용으로 저장을 해준다. Node들 전부를 출력을 해 주어야 함으로 재귀를 이용을 하여서 다른 Node들을 살펴볼수 있게 해주었다. 만약 다음 Node가 존재 하지 않는다면 현재 result 만을 반환을 해주고 다음 Node가 존재한다면 현재 result의 내용 뒤에 다음 Node의 내용을 덧 붙여 준다. 그렇게 하면 모든 Node를 방문을 하면서 type과 value들을 출력을 해줄수 있다.

## def _create_node(self, token):

```
def _create_node(self, token):
    # 토큰을 Node로 만듦
    if token is None:
        return None
    elif token.type is CuteType.INT:
        return Node(TokenType.INT,  token.lexeme)
    elif token.type is CuteType.ID:
        return Node(TokenType.ID,   token.lexeme)
    elif token.type is CuteType.L_PAREN:
        return Node(TokenType.LIST, self._parse_expr_list())
    elif token.type is CuteType.R_PAREN:
        return None
    elif token.type is CuteType.TRUE:
        return Node(TokenType.TRUE)
    elif token.type is CuteType.FALSE:
        return Node(TokenType.FALSE)
    elif token.type is CuteType.PLUS:
        return Node(TokenType.PLUS)
    elif token.type is CuteType.MINUS:
        return Node(TokenType.MINUS)
    elif token.type is CuteType.TIMES:
        return Node(TokenType.TIMES)
    elif token.type is CuteType.DIV:
        return Node(TokenType.DIV)
    elif token.type is CuteType.LT:
        return Node(TokenType.LT)
    elif token.type is CuteType.RT:
        return Node(TokenType.RT)

    elif token.type is CuteType.EQ:
        return Node(TokenType.EQ)
    elif token.type is CuteType.DEFINE:
        return Node(TokenType.DEFINE)
    elif token.type is CuteType.COND:
        return Node(TokenType.COND)
    elif token.type is CuteType.NOT:
        return Node(TokenType.NOT)
    elif token.type is CuteType.CAR:
        return Node(TokenType.TRUE)
    elif token.type is CuteType.CONS:
        return Node(TokenType.CONS)
    elif token.type is CuteType.EQ_Q:
        return Node(TokenType.EQ_Q)
    elif token.type is CuteType.ATOM_Q:
        return Node(TokenType.ATOM_Q)
    elif token.type is CuteType.NULL_Q:
        return Node(TokenType.NULL_Q)
    elif token.type is CuteType.QUOTE:
        return Node(TokenType.QUOTE)
    else:
        return None
```

## 문제 해결 방법

- create_node() 함수는 CuteType에 대응이 되는 Node를 만들어서 반환을 해주는 함수이다. INT와 ID만 Node에 type과 value값을 저장을 하고 나머지들은 type 만을 저장을 한다. 이 중에서 L_PAREB 일 때에는 _parse_expr_list()함수를 호출을 하여 Node를 이용을 하여 연결리스트를 구성을 한다.

## 느낀점

- 이번 과제는 지금까지 해 왔던 과제들의 연장선이어서 저번 과제들에 대한 이해들이 필요했다고 생각을 하였다. 첫 번째 과제에서 사용을 한 재귀를 잘 알고있어야 하였고 저번 과제에서 parsing을 한 결과에 대해서도 이해를 하고 있어야 하였다 하지만 이번 과제에서 가장 놀라웠던 것은 Node의 next에 Node가 들어가는 것 뿐만이 아니라 value에도 Node의 객체를 넣음으로 파이썬의 간편함을 새삼 다시 느낄수 있었다.